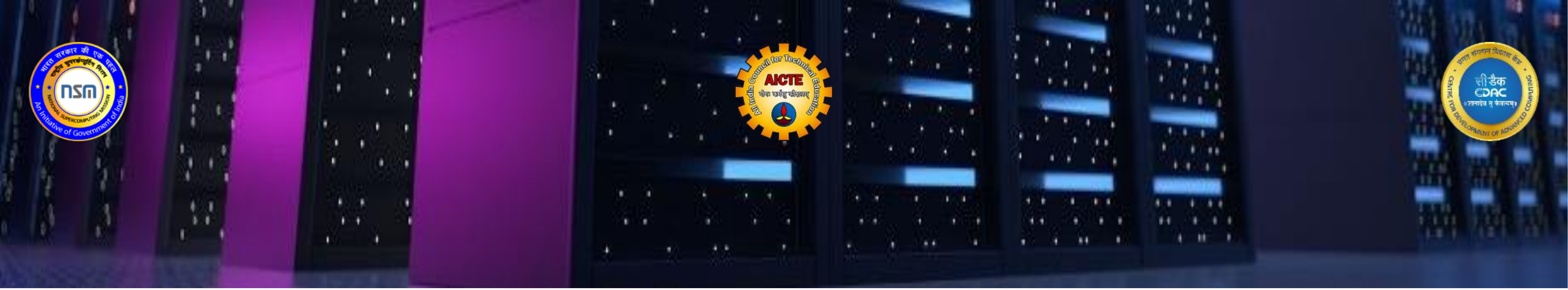


HPC Master Trainer Workshop – February 2024

MPI Collective Communication

Sandeep Agrawal

sandeepa@cdac.in

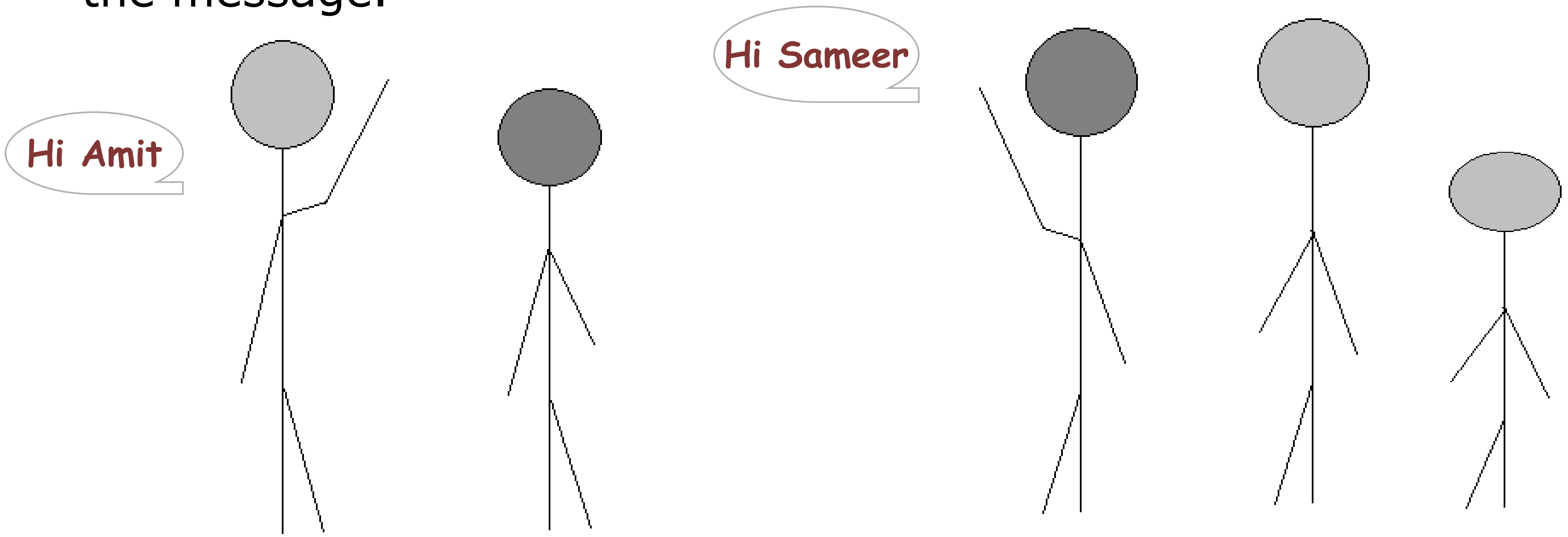
Agenda

- Collective Communication
- Types Of Collective Communication
- Collective Communication Routines
- Collective Communication II Routines
- Some Advance MPI Features

Point To Point Communication

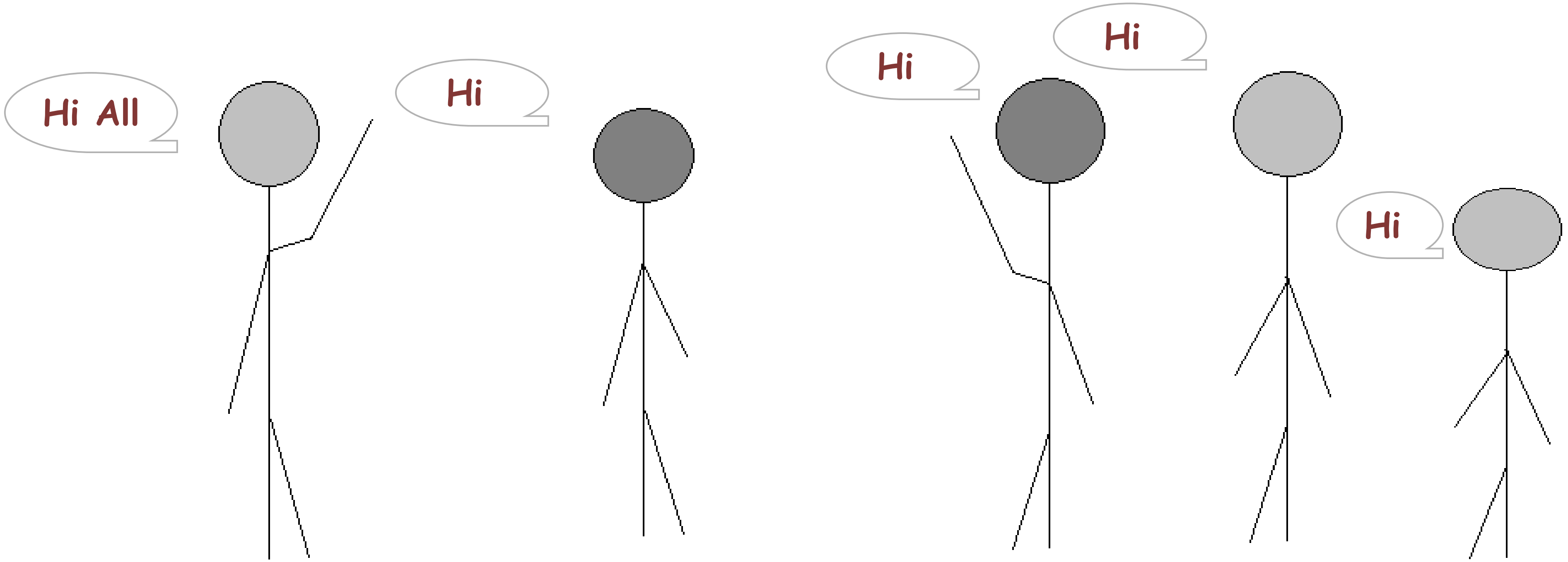
Simplest form of message communication

- Message is sent from a sending process to a receiving process only these two process need to know anything about the message.



Collective Communication

- Collective communication must involve all processes in the scope of a communicator.
- Collective Participation in solving the problem.



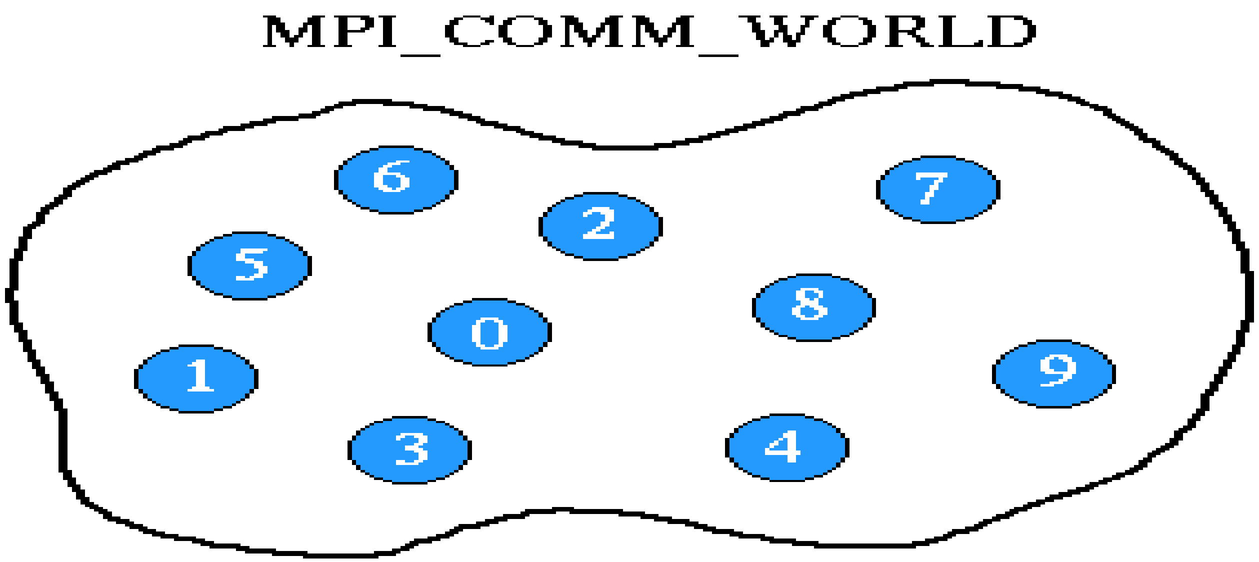
Characteristics

MPI collective communication routines differ in many ways from MPI point-to-point communication routines

- Involves coordinated communication within a *group* of processes identified by an MPI communicator.
- Substitute for a more complex sequence of point-to-point calls.
- All routines block until they are locally complete.
- In some cases, a *root* process originates or receives all data.
- Amount of data sent must exactly match amount of data specified by receiver.
- No message tags are needed.

Communicators and Groups

- MPI uses objects called communicators and groups to define which collection of processes may communicate with each other. Most MPI routines require you to specify a communicator as an argument.



Collective Communication

- Collective communication must involve all processes in the scope of a communicator. All processes are by default, members in the communicator MPI_COMM_WORLD

- It is the programmer's responsibility to ensure that all processes within a communicator participate in any collective operations.

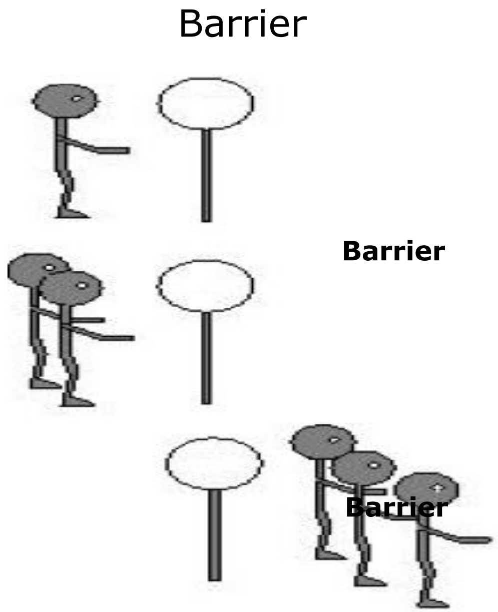
Types of Collective Operations

- Synchronization   Processes wait until all members of the group have reached the synchronization point.
- Data Movement   broadcast, scatter/gather, all to all.
- Collective Computation (reductions) - one member of the group collects data from the other members and performs an operation (min, max, add, multiply, etc.) on that data.

Collective Communication Routines

Barrier

- Creates a barrier synchronization in a group.
- Each task, when reaching the MPI_Barrier call, blocks until all tasks in the group reach the same MPI_Barrier call.



MPI_Barrier (comm) - C Program

MPI_BARRIER (comm,ierr) - Fortran Program

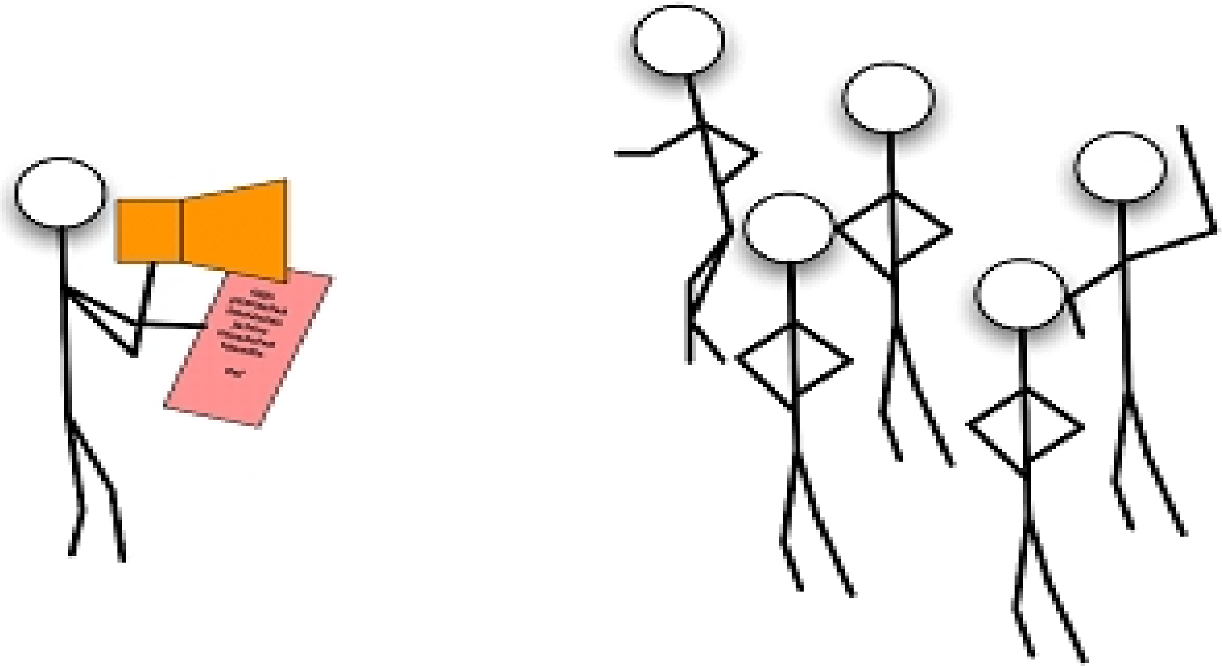
Data Movement

MPI provides three types of collective data movement routines

- Broadcast
- Gather
- Scatter

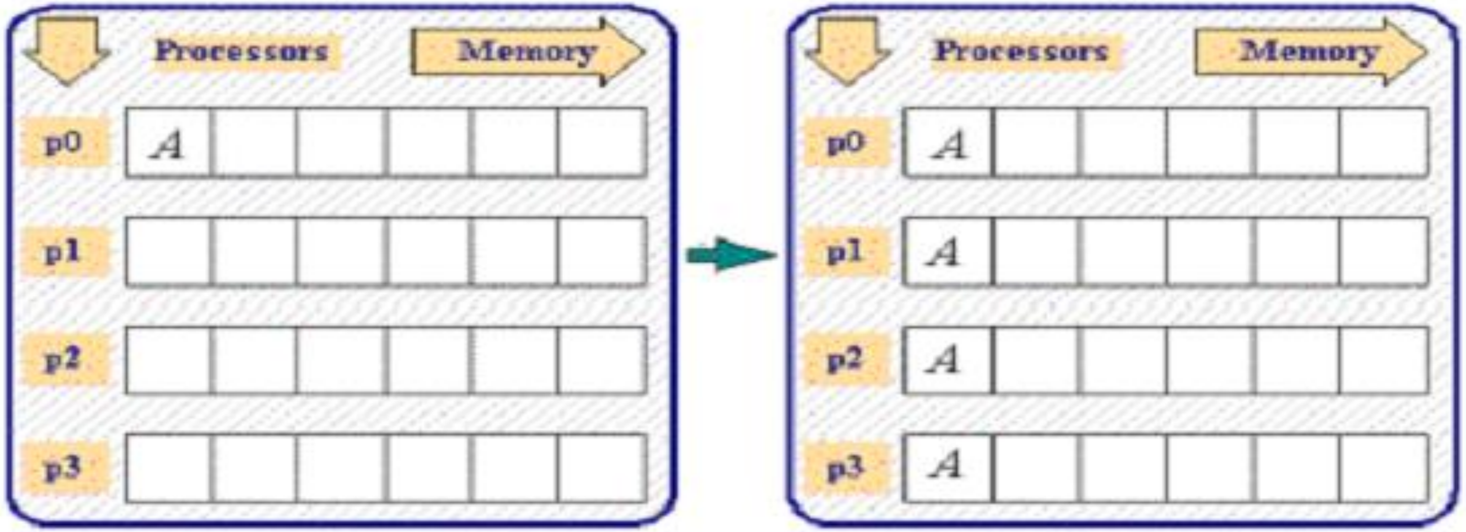
Broadcast

- Broadcasts (sends) a message from the process with rank "root" to all other processes in the group.



Broadcast

- Sends data stored in buffer buf of process source to all the other processes in the group comm.



Broadcast

int MPI_Bcast(void *buf, int count, MPI_Datatype datatype, int source, MPI_Comm comm) - C Program

buf : the address of the send buffer.

count : the number of elements sent to each process.

datatype is MPI defined constant indicating the data type of the elements in the buffer.

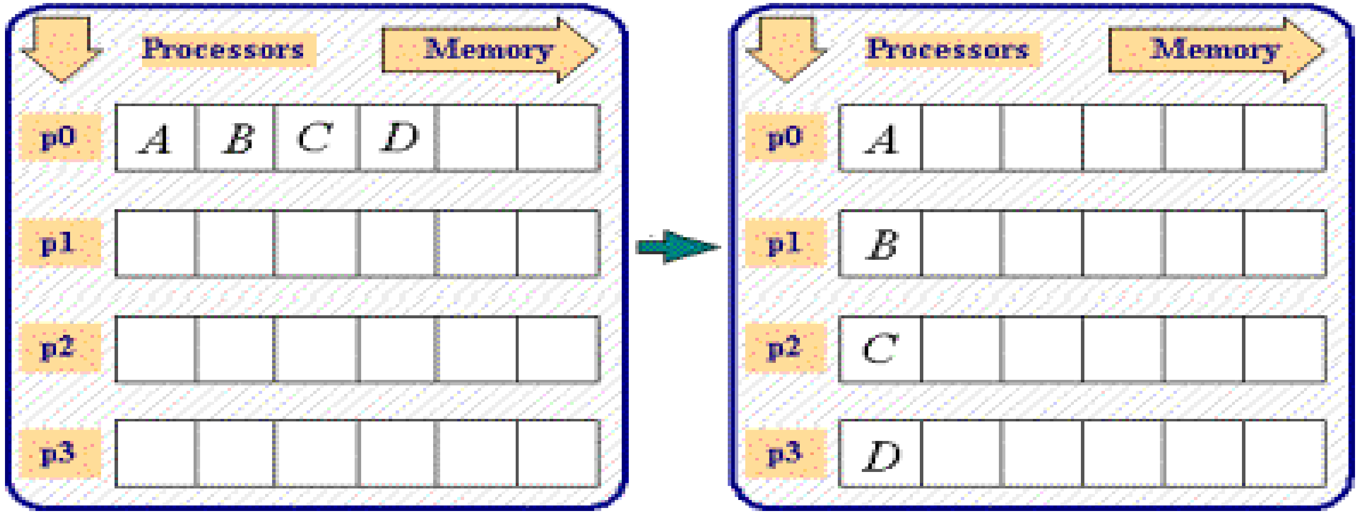
root : is an integer indicating the rank of broadcast

comm : the communicator.

Scatter

- Distributes distinct messages from a single source task to each task in the group. if one wants to distribute the data into n equal segments, where the ith segment is sent to the ith process in the group which has n processes.

MPI_Scatter(&sendbuf,sendcnt,sendtype,&recvbuf ,recvcnt,recvtype,root,comm)



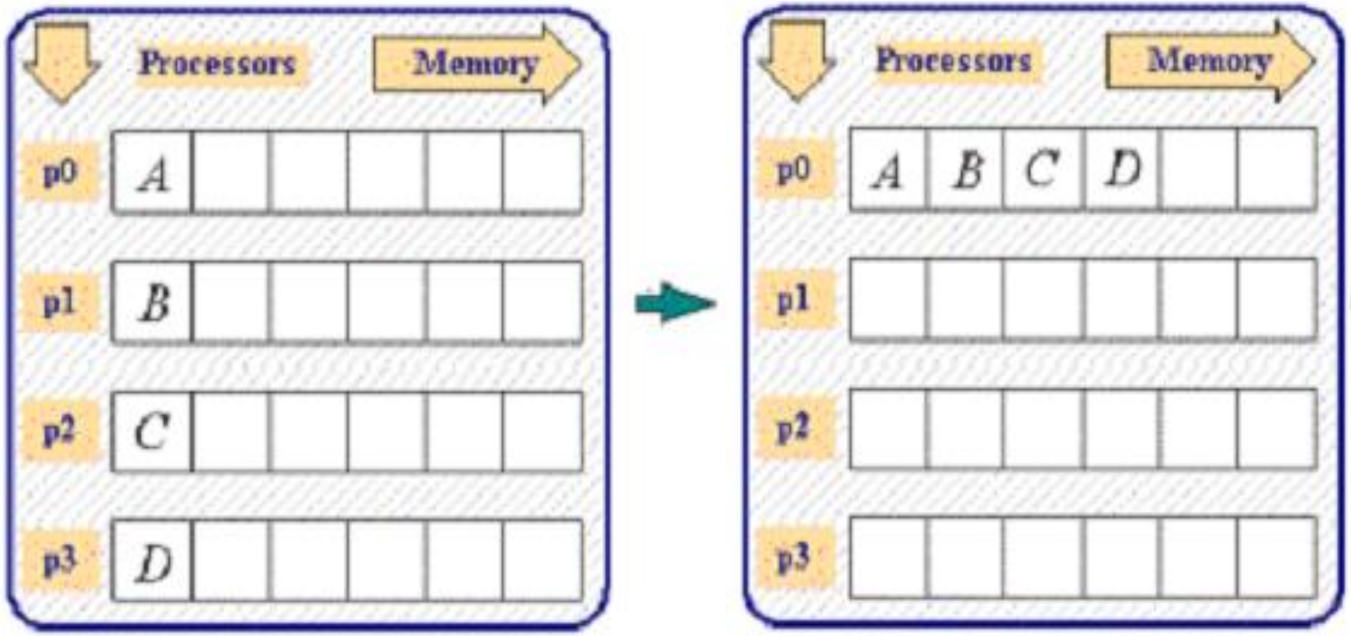
Scatter

MPI_Scatter(&sendbuf,sendcnt,sendtype,&recvbuf,recvcnt,recvtype,root,comm)

sendbuf	:	the address of the send buffer.
sendcnt	:	the number of elements sent to each process.
sendtype	:	the data type of the send buffer elements.
recvbuf	:	the address of the receive buffer.
recvcnt	:	the number of elements in the receive buffer.
recvtype	:	the data type of the receive buffer elements.
root	:	the rank of the sending process.
comm	:	the communicator.

Gather

- Gathers distinct messages from each task in the group to a single destination task. This routine is the reverse operation of MPI_Scatter.



Gather

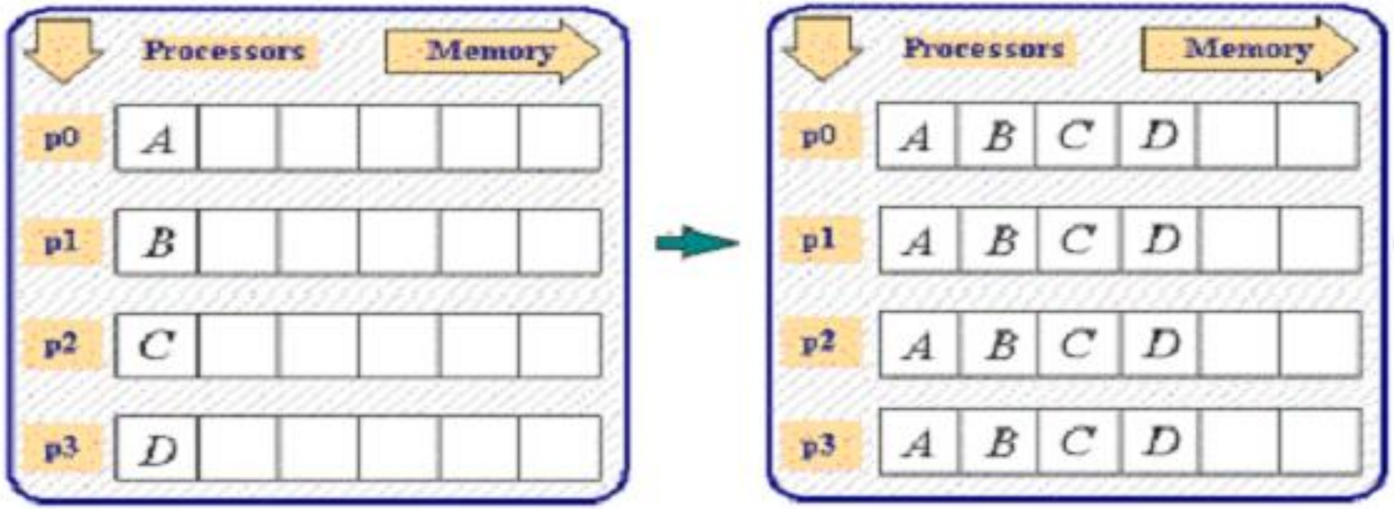
MPI_Gather(&sendbuf,sendcnt,sendtype,&recvbuf,recvcount,recvtype,root,comm)

sendbuf	:	the address of the send buffer.
sendcnt	:	the number of elements in the send
sendtype	:	the data type of the send buffer elements
recvbuf	:	the starting address of the receive buffer.
recvcnt	:	the number of elements for any single receive.
recvtype	:	the data type of the receive buffer elements.
root	:	the rank of the receiving process.
comm	:	the communicator.

AllGather

- Concatenation of data to all tasks in a group. Each rank in the group in effect performs a one – to – all broadcast.

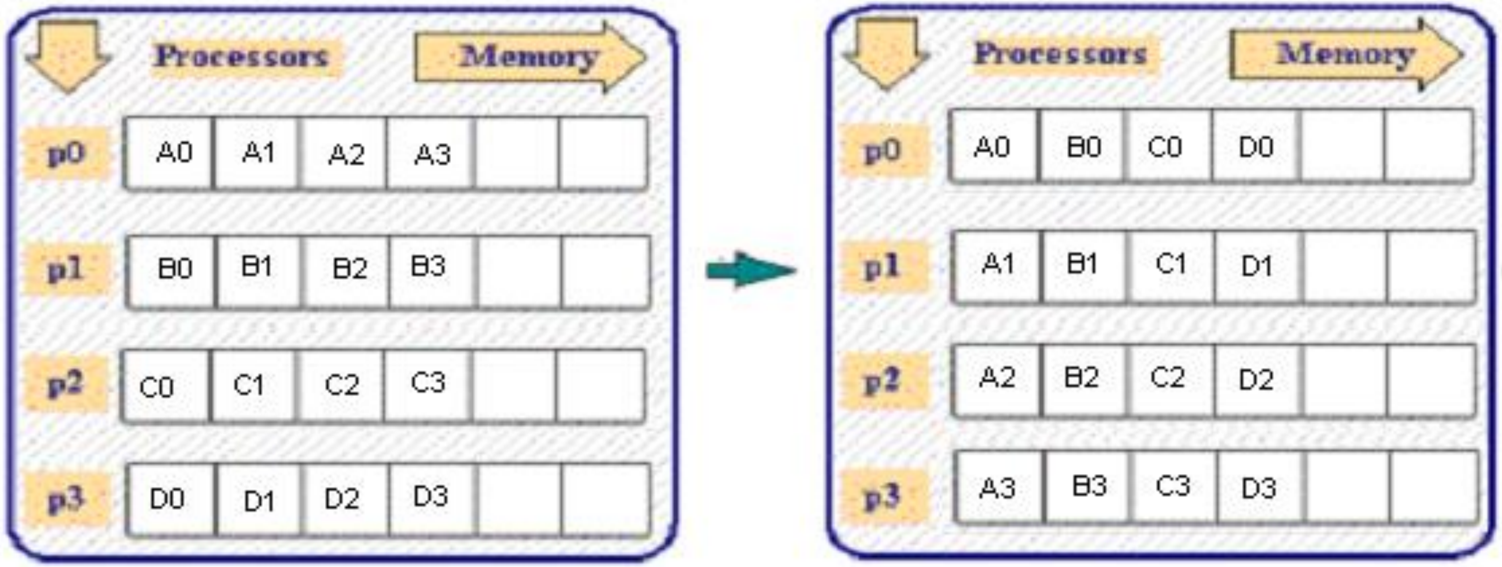
int MPI_Allgather(void *sendbuf, int sendcount,MPI_Datatype senddatatype, void*recvbuf, int recvcount,MPI_Datatype recvdatatype, MPI_Comm comm)



All-to-All

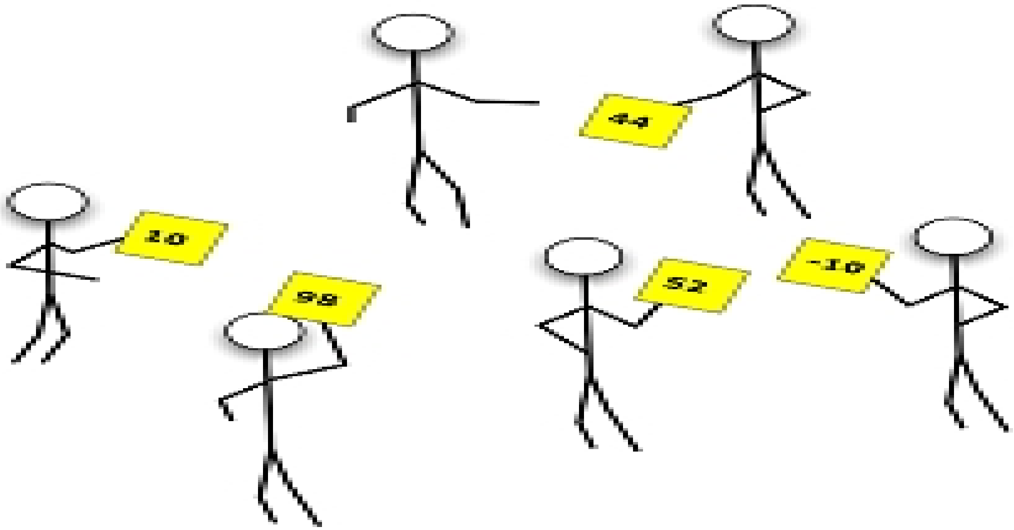
Each process sends a different portion of sendbuf to each other process (incl. itself)

- recvbuf of target process stores data in rank order
- sendcount specifies no. of elements sent to each process



Reduce

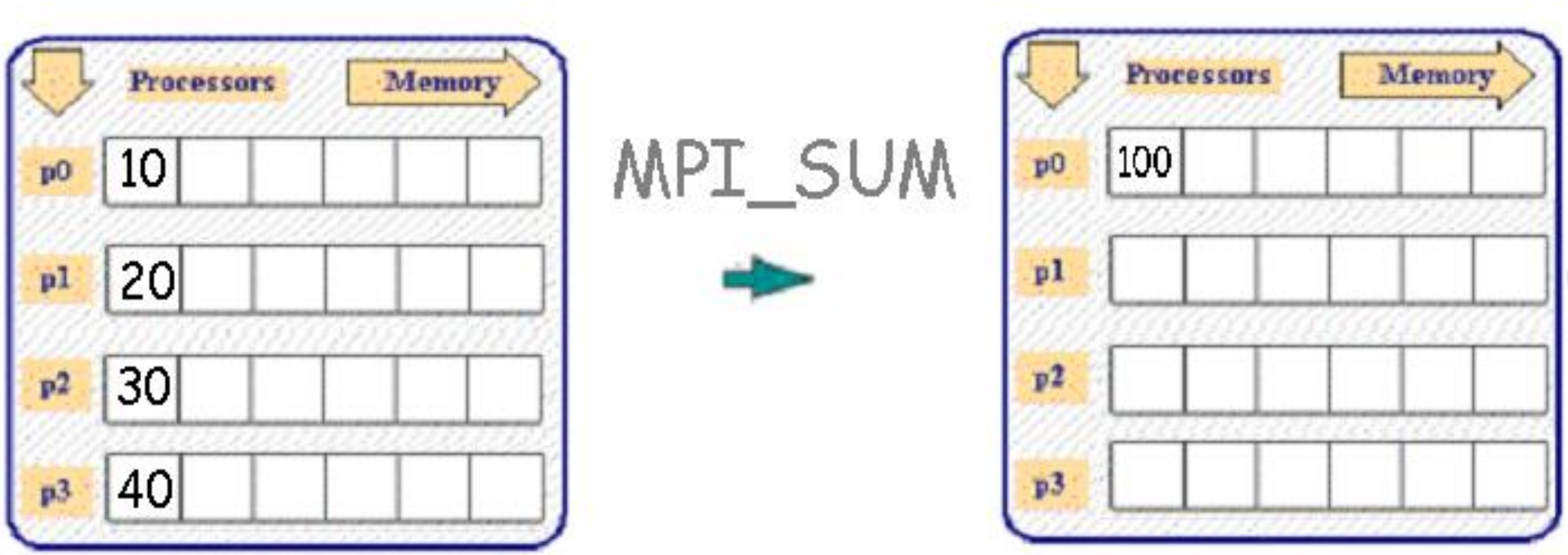
- Used to combine partial results from all processors
- Result returned to root processor
- Several types of operations available
- Works on single elements and arrays



Reduce

MPI_Reduce (&sendbuf,&recvbuf,count,datatype,op,root,comm)

MPI_REDUCE (sendbuf,recvbuf,count,datatype,op,root,comm,ierr)



Reduction operations

Operation	Meaning	Datatypes
MPI_MAX	Maximum	C integers and floating point
MPI_MIN	Minimum	C integers and floating point
MPI_SUM	Sum	C integers and floating point
MPI_PROD	Product	C integers and floating point
MPI_LAND	Logical AND	C integers
MPI_BAND	Bit-wise AND	C integers and byte
...	...	...

MPI Collective Communication

The gather, scatter, allgather, and alltoall routines have variable data versions. For their variable data versions, each process can send and/or receive a different number of elements.

- MPI_Scatterv
- MPI_Gatherv
- MPI_Allgatherv
- MPI_Alltoallv

What does the "v" stand for?

varying – size,relative location of the messages.

Summary

P0	P1	P2*	P3	Function	P0	P1	P2	P3
a	b	c	d	MPI_Gather			a,b,c,d	
a	b	c	d	MPI_Allgather	a,b,c,d	a,b,c,d	a,b,c,d	a,b,c,d
		a,b,c,d		MPI_Scatter	a	b	c	d
a,b,c,d	e,f,g,h	i,j,k,l	m,n,o,p	MPI_AlltoAll	a,e,i,m	b,f,j,n	c,g,k,o	d,h,l,p
		b		MPI_Bcast	b	b	b	b
SBuf	SBuf	SBuf	SBuf		RBuf	RBuf	RBuf	RBuf

- Sender/Root process required by MPI_Gather, MPI_Scatter, MPI_Bcast

Some Advance MPI Features

- **Dynamic Processes** - extensions that remove the static process model of MPI. Provides routines to create new processes
- **One-Sided Communications** - provides routines for one directional communications. Include shared memory operations (put/get) and remote operations.
- **Extended Collective Operations** - allows for non-blocking collective operations and application of collective operations to inter-communicators
- **Parallel I/O** - describes MPI support for parallel I/O

Pros and Cons

Pros of MPI Collective Communication

- Substitute for a more complex sequence of point-to-point calls.

- Communications *may, or may not,* be synchronized (implementation dependent).

- No message tags are needed.

Pros and Cons

Cons of MPI Collective Communication

- A great deal of hidden communication takes place with collective communication.

- Performance depends greatly on the particular implementation of MPI

-There may be forced synchronization, not always best to use collective communication

ਧੰਨਵਾਦ
પ્રશંસાવાદઃ
ધન્યવાદ
આભાર
Thank You
धन्यवाद
धन्यवाद
धन्यवाद
ధన్యవాదాలు
നന്ദി